Web technologies

Server-side Scripting Concepts

Lecture slides – Winter term 2010
Alpen-Adria-University Klagenfurt

Christian Koncilia

PHP

PHP

- PHP stands for „PHP: Hypertext Preprocessor“
- Developed in 1995
- Basically, a PHP file is a HTML file with some PHP code inside PHP-delimiters:

```
<html>
  <body>
    <?php
      if (1 == 1) {
        echo "Hello World";
      } else {
        echo "You should never see me!";
      }
    ?>
  </body>
</html>
```

PHP

- Since PHP 4.0, PHP is an OO language
- PHPs „bad“ history:
  - Initially, PHP stood for „Personal Home Page“ and was designed as simple scripting language
  - Including code in HTML was a bad idea (and still is)
  - Nowadays, PHP is a full OO language and still THE web programming language (PHP web apps: Facebook, Typo3, Wordpress, Mediawiki, …)
  - Therefore: never ever write PHP code like the one you saw on the last slide!

PHP

```
<html>
  <body>
    <?php
       echo "Hello World";
    ?>
  </body>
</html>
```

This PHP file will be converted by PHP into the following result which will be returned to the client:

```
<html>
  <body>
    Hello World
  </body>
</html>
```

PHP Basics

php_basics.php

```
$i = 1;
if (1 == $i) {
  echo "i is 1<br>";
} else {
  echo "i is not 1<br>";
}

switch ($i) {
  case 1:
  case 2:
  case 3: echo "i is 1 or 2 or 3<br>"; break;
  case 4: echo "i is 4<br>"; break;
  default: echo "0 > i > 4<br>";

}

while ($i < 4) {
  echo "i is $i<br>";
  $i += 1;   // equal to "$i = $i + 1"
}
```

```
do {
  echo "i is $i<br>";
  $i--; // equal to "$i -= 1" or "$i = $i - 1"
} while ($i > 0);

echo (0==$i ? "i is again 0" : "i is not 0");
```

- **Output**

```
i is 1
i is 1 or 2 or 3
i is 1
i is 2
i is 3
i is 4
i is 3
i is 2
i is 1
i is again 0
```

PHP Variables / Types

- Variables always start with a $

```
$i = "5,- Euro";
$j = 15;
```

- Variables are case sensitive

```
$testVar = "5,- Euro";
$testvar = 3;
$Testvar = 18.03;
```

PHP Variables / Types

php_variables.php

- PHP is not typesafe

```
$i = 1;
$j = 0;
$s = 'I am a string';
echo $i . "<br>";    // . concatenates two strings
echo $s . "<br>";
$j = $i + $i + $s;
echo $j . "<br>";
$i = $s;
echo $i . "<br>";
```

- Output

```
1
I am a string
2
I am a string
```

PHP Variables / Types

- What´s the output of:

```
$i = "5,- Euro";
$j = "4,30 Euro";
echo $i + $j;
```

```
$i = "5.- Euro";
$j = "4.30 Euro";
echo $i + $j;
```

PHP Strings

php_strings.php

- How to declare a string:

```
$s1 = 'I am a string<br>';
$s2 = "I am a string too<br>";
$s3 = <<<TEST
Gues
what
I
am...<br>
TEST;

echo $s1;
echo $s2;
echo $s3;
```

PHP Strings

php_strings2.php

- String usage:

```
$s1 = '"I am a string" in a string<br>';
$s2 = "'I am a string' in a string<br>";
$i = 15;
$s3 = 'You have to pay $i Euros<br>';
$s4 = "You have to pay $i Euros<br>";
echo $s1 . $s2 . $s3 . $s4;
```

- Output

```
"I am a string" in a string
'I am a string' in a string
You have to pay $i Euros
You have to pay 15 Euros
```

PHP Functions

php_functions.php

```
// $i = 1 defines an initial value
// if called without parameter
function dosomething($i = 1) {
    echo ++$i;
    echo "<br>";
}

dosomething(5);
dosomething();
```

PHP Functions

- Call by Reference / Value.
- Default = Call by Value.

```
function dosomething(&$i) {
   $i++;
}

$x = 1;
dosomething($x);
echo $x;
```

PHP Arrays

- There are no collections in PHP (see Java: Stack, Vector, Dictionary, Hashtable, …)
- However, PHP Arrays are extremely powerful
- Arrays in PHP are Key/Value Pairs
- Key = Integer or String

PHP Arrays

php_array.php

```
$students = array('Tim', 'John', 'Chris', 'Steve');
$students[] = 'Jim'; // add Jim to the array
echo $students[0] . "<br>";
foreach ($students as $key => $value) {
   echo $key . ": " . $value . "<br>";
}
```

- **Output**

```
Tim
0: Tim
1: John
2: Chris
3: Steve
4: Jim
```

PHP Arrays

php_array2.php

```
$student_grades = array('Tim' => 1.4,
                        'John' => 2,
                        'Chris' => 4.3,
                        'Steve' => 3.0);
echo $student_grades[0] . "<br>";
foreach ($student_grades as $key => $value) {
   echo $key . ": " . $value . "<br>";
}
```

Output

```
Notice: Undefined offset: 0 in …\php_array2.php on line 11

Tim: 1.4
John: 2
Chris: 4.3
Steve: 3
```

PHP Arrays

php_array3.php

```
$students = array('Tim' => 'Tim Smith',
                  'John',
                  'Chris' => 'Chris Turing',
                  'Steve SplitPersonality' => array('Steve Nice', 'Steve Bad'));
echo $students[0] . "<br>";
foreach ($students as $key => $value) {
   echo $key . ": ";
   if (is_array($value)) {
     foreach ($value as $v) {
       echo $v . " ";
     }
   } else {
     echo $value . "<br>";
   }
}
```

Output

```
John
Tim: Tim Smith
0: John
Chris: Chris Turing
Steve SplitPersonality: Steve Nice Steve Bad
```

PHP Array Functions

- See http://www.php.net/manual/de/ref.array.php
- explode(…) `"A,B,C" -> ["A","B","C"]`
- implode(…) `["A","B","C"]->"A,B,C"`
- count(…)
- asort(…)
- in_array(…)
- array_key_exists(…)
- …

PHP require, require_once

php_require.php

php_require_test.php

- Includes a file into another file

Php_require_test.php

```
<?php
function printGrades($a) {
   foreach ($a as $k => $v) {
       echo $k . ": " . $v . "<br>";
   }
}
?>
```

Php_require.php

```
require('php_require_test.php');
$student_grades = array('Tim' => 1.1, 'John' => 3.2);
$student_grades['Jim'] = 4.0; // add Jim to the array

printGrades($student_grades);
```

PHP require, require_once

- require_once works like require except that PHP will check if the file has already been included, and if so, not include (require) it again

PHP Parameters

- Parameters (i.e. user input) can be passed to the server using GET or POST (or COOKIE)
- We are going to discuss cookies later

PHP Parameters

```
<form method="GET|POST" action="test.html">
       First name: <input name="firstname"><br>
       Last name:  <input name="lastname"><br>
       <input type=submit>
</form>
```

	GET	POST
form parameters	encoded in URL of HTTP-header, e.g.: http://localhost/test.html?firstname=horst&lastname=pichler	key-value pairs in HTTP-body, e.g. firstname=horst lastame=pichler
parameter passing to the script	passed via environment variable QUERYSTRING	passed via stdin, length of body in CONTENT_LENGTH (stream contains no EOF) and CONTENT_TYPE
constraints	max length of URL (length depends on browser and server)	none
browser history	URL is stored in browser history	not stored (resubmit necessary)
usage	e.g., for simple search-forms	e.g., for file-upload forms

PHP Suberglobals

- PHP has some superglobals defined.
- Superglobals are variables that the web server or PHP fills with values.
- Superglobals are accessible in all scopes
- Superglobals:

```
$_SERVER        $_REQUEST
$_SESSION       $_ENV
$_GET           $_FILES
$_POST          $_COOKIE
```

PHP Parameters

- Get POST or GET parameters in PHP
- $_GET
  - Returns an associative array of all variables passed via URL
- $_POST
  - Returns an associative array of all variables passed via HTTP POST method

PHP Parameters

php_get.php

```
$vars = $_GET;
var_dump($vars);
```

- **Call:**
- http://dev.webtech2010.com/chapter - 3 php/php_get.php?i=1&name=Christian
- **Output**

```
array(2) {["i"]=> string(1) "1"
          ["name"]=> string(9) "Christian" }
```

PHP Parameters

php_get2.php

```
<form action="php_get2.php" method="GET">
   <input name="yourname" type="text">
   <input type="submit" value="send">
</form>
<?php
   $vars = $_GET;
   if (array_key_exists('yourname', $vars)) {
      echo "your name is: " . $vars['yourname'];
   } else {
      echo "no name submitted yet";
   }
?>
```

- After clicking on the submit-button, check the URL!

PHP Parameters

php_get3.php

```
<form action="php_get3.php" method="POST">
    <input name="yourname" type="text">
    <input type="submit" value="send">
</form>
<?php
    $vars = $_POST;
    if (array_key_exists('yourname', $vars)) {
        echo "your name is: " . $vars['yourname'];
    } else {
        echo "no name submitted yet";
    }
?>
```

- After clicking on the submit-button, check the URL!

PHP Cookies

- Cookies are small chunks of data stored on the client!
- Set a cookie:

```
setcookie("TestCookie", $value);
```

- Set a cookie with expire date:

```
setcookie("TestCookie", $value, $time);
```

- Where $time is a Unix Timestamp (number of seconds since Januar 1 1970 00:00:00 GMT). Use time() function to get Unix Timestamp of „now“
- Force client-browser to delete a cookie: Set timestamp to a value in the past!

```
setcookie ("TestCookie", "", time() - 9999);
```

PHP Cookies

php_cookies.php

```
echo "Run me twice!!!<br><br>";
$var = $_COOKIE;
var_dump($var);
$value = '35';
setcookie("TestCookie", $value);
setcookie("TestCookieWithTimeout3", $value, time()+ 3);
```

- Output (first run):

```
array(0) { }
```

- Output (second run):

```
array(1) { ["TestCookie"]=> string(2) "35" }
```

Or (depending on fast you reload the page)

```
array(2) { ["TestCookie"]=> string(2) "35"
["TestCookieWithTimeout3"]=> string(2) "35" }
```

Sessions

- In all web programming languages, a web page is a web page is a web page…
- If you surft from \page1.html to \page2.html these are two unique requests.
- The server doesn‘t know anything about the fact, that both pages are visited by the same user.
- Sessions are the technique used to logically group several requests into a „group“ (called session)
- If you start a session, the server will know that it‘s still the same user who surfed from \page1.html to \page2.html

Sessions

- HTTP itself is „stateless“
  - no state stored on the server between requests from the same client
- but many web apps are stateful
  - necessary to connect requests from the same user / browser / browser-window, e.g.
    - LV-online: displaying lists for a specific logged in user
    - amazon shopping carts
- session
  - multiple requests performed in a stateful context
- session tracking
  - techqnique that allows sessions in stateless environments

Session Tracking / Handling

- User surfs to http://dev.webtech2010.com
- server (on 1st request / if no sessionID stored on client)
  - generates unique session id, which is mapped to ...
  - ... a session-object
    - stored in memory (lost on shutdown), files, database
    - can contain anything (list of articles, game state, counters, ...)
  - session id is added to the response (how? see next page)
- from now on
  - each subsequent request from the same user (browser) must contain the session id ...
  - ... which is used by the server to map to the session-object
- No data gets stored on the client, except SessionID !!!

Session Tracking with Cookies

1. server creates a cookie with session-id on first request
2. server maps id to a new user-specific session object
3. the session-id is sent to the client with the first response
4. and automatically added by the browser on each further request (to the same address/domain/…)
5. server receives request + cookie with session-id
6. server maps session-id to session-object

- Potential problems
  - users may disallow the usage of cookies in their browsers

Session Tracking with URL-rewrite

- Basic idea
  - Server adds the session-id to all links the user can follow

    http://server/MyServlet

    is changed to

    http://server/MyServlet?sessionid=123
  - session-id must be dynamically added
    - functionality usually offered by scripting frameworks

- Pro
    - simple, works with every browser (no cookies required)
- Contra
    - all URLs in response pages must add session-id
    - URL displayed in browser is the rewritten URL

Sessions Tracking: Hidden Form Fields

- In HTML, one can define "hidden" fields in a form

```
<INPUT TYPE="HIDDEN" NAME="sessionid" VALUE="123">
```

- These fields are not visible and cannot be changed by the client
- Usage:
  - server creates a session-object for each client and generates a unique ID
  - When HTML documents are created and sent back, the hidden form field is automatically generated containing the actual ID
  - Upon form submit, the session ID is automatically sent back to the server
  - The server can associate this call with an already existing session
- Pro:
  - Simple, works with every browser (no cookies required)
- Contra:
  - Form must be added to all pages
  - Form must be submitted at each request to the server!!!

Session Tracking

- When does a session end?
  - when browser(window) is closed
  - when user logs out
  - when the session times out
    - after configured timespan without further request

- Popular session tracking techniques
  - session tracking based on cookies
  - URL-rewrite
  - hidden form-fields

PHP Sessions

php_session.php

```
<form action="php_session.php" method="POST">
   <input name="username" type="text">
   <input name="password" type="password">
   <input type="submit" value="Send">
</form>
<?php
   session_start(); // thats the magic function...

   $username = (isset($_POST['username']) ? $_POST['username'] : null);
   $password = (isset($_POST['password']) ? $_POST['password'] : null);

   if ("koncilia" == $username && "root" == $password) {
      $_SESSION['username'] = $username;
      echo "Hi Christian! You are now logged in.<br><br>";
      echo "<a href='php_session_cont.php'>Continue to next page...</a>";
   } else {
      if (null != $username) {
         echo "Wrong username / password";
      } else {
         echo "no name submitted yet";
      }
   }
?>
```

PHP Sessions

php_session_cont.php

```
<?php
   session_start(); // thats the magic function...

   if (isset($_SESSION['username'])) {
      echo "I still know that you are " . $_SESSION['username'] . "...<br>";
   } else {
      echo "I'm sorry but you are NOT logged in!...<br>";
   }
?>
```

PHP $_SERVER

php_server.php

- $_SERVER is an array which contains information about the request.
- Values are set by the web server (i.e. Apache).
- Important keys:
  - HTTP_HOST (def.webtech2010.com)
  - HTTP_USER_AGENT (Browser used)
  - HTTP_ACCEPT_LANGUAGE (Browser language)
- Example:

```
foreach ($_SERVER as $k => $v) {
   echo "<span style='color:red;'>" . $k . "</span>: " . $v . "<br>";
}
```